Sveučilište J.J. Strossmayera u Osijeku

Filozofski fakultet

Odsjek za informacijske znanosti

Preddiplomski studij informatologije

Matija Vavetić

Pozadinske tehnologije u radu s označiteljskim jezicima

Završni rad

Mentor izv. prof. dr. sc. Boris Bosančić

Osijek, 2019.
Sveučilište J.J. Strossmayera u Osijeku

Filozofski fakultet

Odsjek za informacijske znanosti

Preddiplomski studij informatologije

Matija Vavetić

POZADINSKE TEHNOLOGIJE U RADU S OZNAČITELJSKIM JEZICIMA

Završni rad

Društvene znanosti, informacijske i komunikacijske znanosti, informacijski sustavi i informatologija

Mentor izv. prof. dr. sc. Boris Bosančić

Osijek, 2019.

Prilog: Izjava o akademskoj čestitosti i o suglasnosti za javno objavljivanje

Obveza je studenta da donju Izjavu vlastoručno potpiše i umetne kao treću stranicu završnog odnosno diplomskog rada.

IZJAVA

Izjavljujem s punom materijalnom i moralnom odgovornošću da sam ovaj rad samostalno napravio te da u njemu nema kopiranih ili prepisanih dijelova teksta tuđih radova, a da nisu označeni kao citati s napisanim izvorom odakle su preneseni.
Svojim vlastoručnim potpisom potvrđujem da sam suglasan da Filozofski fakultet Osijek trajno pohrani i javno objavi ovaj moj rad u internetskoj bazi završnih i diplomskih radova knjižnice Filozofskog fakulteta Osijek, knjižnice Sveučilišta Josipa Jurja Strossmayera u Osijeku i Nacionalne i sveučilišne knjižnice u Zagrebu.

U Osijeku, datum 10.3.2019.

Matija Vučetić, 0122223541
ime i prezime studenta, JMBAG

Sadržaj

Sažetak

Svrha rada je prikazati mogućnosti zajedničkog djelovanja označiteljskih jezika i pozadinskih tehnologija. U skladu sa svrhom, kreirana je praktična aplikacija u obliku API-ja temeljena na PHP programskom i XML označiteljskom jeziku. U prvom dijelu rada izložen je povijesni razvoj označiteljskih jezika, te je detaljno opisan XML kao najvažniji označiteljski jezik. Isto tako, opisana je povijest i razvoj programskih jezika, odgovoreno na pitanje što su ustvari pozadinske tehnologije te navedeni odgovarajući primjeri. Nakon toga, u drugom dijelu rada izložen je detaljan pregled praktične aplikacije naziva *TestData*. Glavna primjena aplikacije je stvaranje testnih podataka o izmišljenim osobama u odgovarajućoj bazi podataka. Prednost ovakve vrste API-ja jest ušteda vremena developerima u početnim fazama razvoja softvera kada su im potrebni podaci za testiranje. Cilj aplikacije je prikazivanje međusobnog i simultanog djelovanja označiteljskih jezika i programskih jezika na način da se slanjem zahtjeva bazi podataka kroz aplikaciju dobiju odgovori u XML ili JSON formatu.

Ključne riječi: pozadinske tehnologije, PHP, označiteljski jezici, XML, API

1. Uvod

Svrha ovog rada je prikaz mogućnosti zajedničkog djelovanja označiteljskih jezika i pozadinskih tehnologija. U skladu sa svrhom rada, kao cilj rada napravljena je i praktična aplikacija u obliku API-ja, temeljena na, između ostalih, PHP programskom i XML označiteljskom jeziku putem koje su se te mogućnosti i praktično demonstrirale. Aplikacija je izrađena u obliku API-ja, sučelja baze podataka putem kojeg su se podaci iz baze podataka mogli stvarati, uređivati, brisati i čitati.

Prvo poglavlje bavi se označiteljskim jezicima u kojem se opširnije opisuje povijest i razvoj istih. Opisuje se povijest razvoja *GML*-a (*Generalized Markup Language*), i *GenCode*-a 1960-ih i 1970-ih čijim udruživanjem u 1980-im nastaje *SGML* (*Standard Generalized Markup Language*), iz kojeg su na kraju proizašli označiteljski jezici kao što su XML i HTML. Zatim se opisuju i dvije vrste označiteljskih jezika: proceduralni i opisni. Nadalje, detaljno se opisuje XML kao jedan od najvažnijih i najpopularnijih označiteljskih jezika u mrežnom okruženju. Istaknuta je njegova svrha i posebne namjene, uz poseban naglasak na prikaz XML sintakse. Na kraju se spominje i XML Schema koja predstavlja način definiranja strukture XML dokumenata.

Drugo poglavlje posvećeno je programskim jezicima i pozadinskim tehnologijama. Dan je osvrt na povijest i razvoj programskih jezika, počevši od izuma diferencijalnog stroja, do razvoja Pascal-a iz kojeg je proizišao C jezik, a iz kojeg je pak proizišao C++ i objektno orijentirana paradigma programiranja. Nedugo zatim, nastaju jezici kao što su Java, C# i PHP, koji se smatraju pozadinskim tehnologijama. Zatim je opisan PHP jezik koji se koristio prilikom izrade praktične aplikacije, a isto tako i Laravel, iznimno robusno i brzo razvojno okruženje za PHP jezik.

Treće poglavlje predstavlja praktičnu aplikaciju *TestData* napravljenu s namjerom demonstracije svrhe ovog rada u praksi. Aplikacija predstavlja ustvari API, odnosno sučelje za programiranje aplikacija. Dan je generalni pregled aplikacije, te način i tehnologije koje su korištene prilikom izrade, a isto tako detaljnije je opisana njena funkcionalnost i mogućnosti koje pruža a što je popraćeno i konkretnim primjerima.

2. OZNAČITELJSKI JEZICI

2.1. Povijest i razvoj označiteljskih jezika

Označiteljski jezici (engl. *markup languages*), računalni su jezici koji se koriste za određivanje uputa za označavanje podataka u dokumentu. Drugim riječima, upute za označavanje nisu podaci u dokumentu, oni su podaci o podacima u dokumentu. Prvi računalni program za formatiranje teksta koji je koristio označiteljski princip bio je RUNOFF koji je 1964. razvio J. H. Saltzer. Krajem 60-ih godina prošloga stoljeća dolazi do skraćivanja golemih naredbi u tzv. makroe (engl. *macros*). Makroi označavaju skraćenu verziju naredbe, a naziv su dobili po hijerarhijskoj odrednici koja ih odvaja od naredbi na nižoj razini. Označiteljski jezici nastali su u onom trenutku kada se više nije moglo točno odrediti jesu li makroi samo skraćenice ili predstavljaju posebne identifikatore komponenti. Jedna od glavnih prednosti makroa odnosi se na mogućnost izmjene naredbi za formatiranje pojedinih komponenti teksta.[1] Isto tako, tijekom 1960-ih počinju se u ranoj fazi razvijati makro označiteljski jezik *GML* (*Generalized Markup Language*), na kojem rade C. Goldfarb, E. Mosher i R. Lorie. Osim *GML*-a, Norman Schrapf pokreće *GenCode* projekt, koji je imao za cilj razviti standardni opisni označiteljski jezik za izdavaštvo. Goldfarb nastavlja sa svojim radom na *GML*-u, te se prva inačica objavljuje 1973. Korištenjem GML-a, dokument je označen oznakama koje određuju što je tekst u kontekstu odlomaka, zaglavlja, popisa, tablica itd. Nedugo nakon toga, dolazi do udruživanja *GenCode*-a i *GML*-a te je 1980. nastao prvi radni nacrt *SGML*-a, a 1986. dolazi do objave njegove prve službene inačice.

Također je bitno spomenuti i *Scribe*, označiteljski jezik koji je dizajnirao i razvio Brian Reid u 1980-ima, i kojim je oslobodio autore od briga oko oblikovanja teksta pružajući im integrirani alat za bibliografiju i upravljanje citatima.[2] Reidov koncept razdvajanja strukture od prezentacije bio je pionirski, te se smatra pretečom različitih jezika i tehnologija kao što su: CSS (*Cascading Style Sheets*), *JavaScript*, a ujedno imao je i velik utjecaj na razvoj SGML-a, iz čega je proizašao XML, koji je danas najkorišteniji označiteljski jezik uz HTML. SGML (*Standard Generalized Markup Language)* jezik je za stvaranje strojno čitljivih definicija elemenata deskriptivnih označiteljskih jezika. Gotovo neizbježna prva zabluda vezana uz SGML je kako je SGML jezik za označivanje dokumenata s unaprijed definiranim nizom

[1] Bosančić, B. Povijest označiteljskih jezika. Označiteljski jezici za prikaz i opis sadržaja. Osijek, 2017. [Predavanje]
[2] Coombs, J. H., Renear, A. H., & DeRose, S. J. Markup systems and the future of scholarly text processing. // Communications of the ACM, 30(11), 1987. str. 933-947. URL: https://doi.org/10.1145/32206.32209 (2019-08-16)

oznaka za označavanje dokumenata. No, unatoč svom nazivu, SGML nije označiteljski jezik u tom smislu i ne sadrži oznake za opis sadržajnih objekata. On je „metajezik“, odnosno jezik za definiranje jezika označitelja dokumenata. Zbunjenost povećava i činjenica da oba označiteljska "metajezika" – SGML i XML (*Extensible Markup Language)*, kao i označiteljski jezici temeljeni na SGML-u i XML-u, kao što su to, npr. HTML i XHTML, također sadrže „označiteljski jezik“ u svom nazivu.[3]

Pojavom weba i HTML-a praksa se mijenja iz korijena. HTML označiteljski jezik baziran je na SGML-u, a prva primjena bila mu je izrada i prikaz mrežnih stranica u mrežnim preglednicima. Kao što je navedeno, HTML je proizašao iz SGML-a, međutim, njihova poveznica bila je izrazito slaba i sadržavala brojne nedostatke. Najbolji primjer loše prakse koja je nastala pojavom HTML-a, je njegovo korištenje u praksi prije nego je objavljen njegov DTD (*Document Type Definition*). Drugim riječima, HTML se počeo koristiti prije nego se počeo koristiti dokument koji je sadržavao njegova „pravila“ tj. unaprijed definiranu strukturu i sastav elemenata. Još jedan nedostatak ovakvog postupanja s novim i modernijim označiteljskim jezikom, a koji pruža niz mogućnosti, upravo je nepromišljeno korištenje tipova elemenata opisnog i proceduralnog označavanja. Zbog nepostojanja posebnog softvera za procesuiranje HTML dokumenta, prikaz HTML dokumenta na ekranu i danas obavljaju mrežni preglednici, što za nedostatak ima različite načine prikaza istog HTML kôda. Stoga je DTD postao nepotreban, jer se HTML morao pridržavati pravila mrežnih preglednika. Svi ti problemi i nedostaci bili su podloga za razvoj XML-a. On je razvijen u okviru *World Wide Web Consortium*-a (W3C), pri čemu je prvi nacrt objavljen 1996., a službena preporuka 1998. Glavni cilj XML-a bio je osigurati da se novi i specijalizirani označiteljski jezici mogu učinkovito koristiti na webu, bez prethodne koordinacije između programera softvera i stvaratelja sadržaja. Jedan od bitnijih ciljeva bio je stvaranje jednostavnije i ograničenije verzije SGML-a, tako da s manje opcija za podršku i drugim pojednostavljenjima, programerima bude lakše razviti SGML/XML softvere, a tada bi i više softvera moglo biti razvijeno i korišteno, a koji bi podržavali individualizirane jezike za označavanje.[4]

Kasnije dolazi do razvoja 1.1 verzije XML-a, koja više ne ovisi o trenutnoj inačici Unicode-a, standarda za predstavljanje znakova i simbola većine pisama svijeta, već uvijek koristi najnoviju inačicu. Kao što je ranije napisano, HTML ne posjeduje poseban softver za procesuiranje HTML dokumenata, dok XML sadrži posebno napisanu aplikaciju za tu namjenu,

[3] Renear, A. H. Text encoding. Nav. dj.
[4] Usp, Isto.

a to je *XML Parser*, koji predstavlja programski paket koji pruža sučelje za klijentske aplikacije za rad s XML dokumentima. Između ostalog, *XML Parser* provjerava ispravnost formata XML dokumenta i potvrđuje njihovu ispravnost. Moderni preglednici imaju ugrađene *XML Parsere*, a njihov cilj je transformirati XML u čitljiv kôd.[5]

2.2. Vrste označiteljskih jezika

2.2.1. Proceduralni označiteljski jezici

Proceduralno označavanje sastoji se od naredbi koje upućuju na koji način bi tekst trebao biti formatiran. Korištenje ovakvog pristupa označavanja daje sustavu za obradu teksta upute kao što su preskakanje određenog broja redova, postavljanje dvostrukog proreda, promjena u jednostruki prored, započinjanje nove stranice ako je ispunjen određen broj redova na stranici, itd.[6] Na sustavima kao što su strojevi za unos teksta, postojali su kontrolni nizovi koje su u terminal unosili korisnici, a koji su bili sadržani u tekstualnim datotekama koje su bile sastavnica prikaza oblika. Sami kontrolni nizovi sastojali su se od brojnih tzv. "označiteljskih elemenata", koji upućuju na početak razvoja proceduralnih označiteljskih jezika, jer su se te oznake smatrale uputama za naknadnu obradu u sustavu za oblikovanje.[7] Ipak, proceduralno označavanje ima svojih nedostataka, a najveći problem odnosi se na činjenicu da logička struktura dokumenta ne ostaje pohranjena nakon što je završeno označavanje dokumenta te da stilsko oblikovanje dokumenta određuje korisnik koji je vršio označavanje. To znači da je za kvalitetno oblikovanje u tom slučaju bilo potrebno znanje o tipografiji. Najpoznatiji proceduralni označiteljski jezik je svakako LaTeX, koji je baziran na već postojećem TeX standardu. Stvorio ga je L. Lamport 1985., a nazvan je sustavom za pripremu dokumenata (engl. *document preparation system*). Glavna pogodnost LaTeX-a je objavljivanje u bilo kojem obliku jer posjeduje unutarnje mehanizme za prikaz prethodno pravilno označenog teksta skupom unaprijed definiranih makronaredbi.

2.2.2. Opisni označiteljski jezici

Označavanje teksta smatra se opisnim kada samo označavanje ne predstavlja oblikovne naredbe ili obradu naredbi, već identificiranje logičke strukture dokumenta. Takva logička struktura predstavlja se opisnim oznakama koje funkcioniraju na način ograničavanja logičkih elemenata

[5] Bosančić, B. Uloga opisnih označiteljskih jezika u razvoju digitalne humanistike. // Libellarium IV, 1(2011), str. 76. URL: https://hrcak.srce.hr/file/136172 (2018-08-18)
[6] Coombs, J.H., Renear, A.H., DeRose, S.J. Nav. dj., str. 936.
[7] Joloboff, V. Trends and Standards in Document Representation. // Text Processing and Document Manipulation: Proceedings of the International Conference University of Nottingham, 1986. str. 108-110

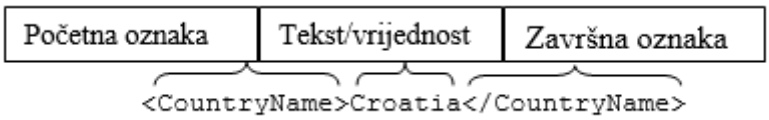
samog dokumenta. Isto tako, elementi sadržavaju leksičke atribute koji predstavljaju nove dodatne informacije. Sami opisni označiteljski jezici predstavljaju skup pravila koja opisno upravljaju označavanjem određenih vrsta dokumenata. Prilikom definiranja opisnog označiteljskog jezika možemo koristiti prethodno navedene „metajezike“ kao što su SGML ili XML, a njihovom uporabom možemo izgraditi sintaksu određenog opisnog označiteljskog jezika, koja nalaže kako u praktičnom smislu provesti označavanje sadržaja određenog dokumenta. Opisni označiteljski jezici u usporedbi s drugim označiteljskim jezicima imaju niz prednosti, ali nemoguće je odrediti koja od njih je najvažnija. Jedna od prednosti je stvaranje teksta na jednostavniji način gdje autor ne mora razmišljati o njegovu formatiranju. Druge prednosti odnose se na strukturno uređivanje teksta, lakše premještanje i brisanje teksta, više načina prikaza identičnog teksta, kompatibilnost s vanjskim uređajima, mogućnost migriranja sadržaja između različitih označiteljskih sustava, a postoji još i niz drugih prednosti.[8] Prema zaključku rada Coombsa, Reneara i DeRosea opisni označiteljski jezici se smatraju „najbolje uopće zamislivim pristupom“ u označavanju teksta, neovisno o području.[9]

2.3. XML

XML predstavlja označiteljski jezik koji je posljednjih godina sve važniji u načinu predstavljanja temeljnih struktura podataka na prenosiv način, odnosno koji podržava razmjenu relevantnih podataka, neovisno o stvarnoj fizičkoj pohrani istih.[10] Također, kao „metajezik“ XML omogućuje kreiranje vlastitog označiteljskog jezika prilagođenog osobnim zahtjevima. Kao i HTML, XML se može koristiti za oblikovanje teksta, ali i za strukturiranje temeljnih podataka. Stoga je jedan od ciljeva XML-a da njegovi dokumenti budu čitljivi ljudima, formalni, sažeti, te jednostavni za stvaranje i lako obradivi. Isto tako, XML bi trebao osigurati podršku u radu različitih aplikacija i biti kompatibilan sa SGML-om.[11] Imajući sve to na umu, lako je uvidjeti da je XML izrazito koristan označiteljski jezik, a zbog tog svog svojstva posjeduje i izrazito karakteristične prednosti kao što su stavljanje naglaska na opisnom, a ne proceduralnom označavanju, razlikovanje koncepata sintaktičke ispravnosti i valjanosti s

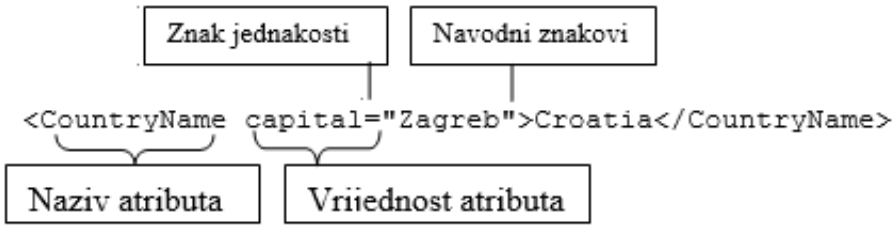
[8] Coombs, J.H., Renear, A.H., DeRose, S.J. Nav. dj., str. 940.
[9] Usp., Isto. str. 946.
[10] Snyder, R. A Practical Introduction to the XML, Extensible Markup Language, By Way of Some Useful Examples. // Proceedings of the 2004 ASCUE Conference, str. 239. URL: https://files.eric.ed.gov/fulltext/ED490125.pdf (2019-08-20)
[11] Blyth, A., Cunliffe, D., Sutherland, I. Security analysis of XML usage and XML parsing. // Computer & Security 22, 6(2003), str. 494-495. URL: https://www.sciencedirect.com/science/article/pii/S0167404803006072?via%3Dihub (2019-08-20)

obzirom na DTD te neovisnost o hardverskom ili softverskom sustavu.[12] Namjena XML-a ogleda se u pohrani, razmjeni i strukturiranju podataka, dok se pod pohranom smatra čuvanje i zaštita podataka. Pored toga, XML se također može koristiti za kodiranje i postavljanje upita prema bazi podataka što omogućuje različite prikaze istog sadržaja te odvajanje sadržaja od njegovog prikaza. Zbog toga, ova vrsta označiteljskih jezika može surađivati s upitnim jezicima kao što je SQL (*Structured Query Language*). Posebna namjena XML-a ogleda se u mogućnosti strukturiranja dokumenta (naslov, poglavlje, potpoglavlje itd.).[13] Kao što je već spomenuto, XML ne posjeduje predefinirane oznake, njih u principu stvara autor dokumenta. Svaki element se sastoji od početne (*start-tag)* i završne oznake (*end-tag*) koje se nalaze unutar znaka manje '<' i znaka veće '>', te ti znakovi zajedno predstavljaju generički identifikator. Unutar početne i završne oznake nalazi se pripadajući tekst, podatak ili vrijednost, što je vidljivo u sljedećem primjeru:



Slika 1. Primjer osnovnog XML elementa sa sastavnim dijelovima

Prilikom imenovanja elemenata potrebno je pridržavati se određenih pravila koja propisuju da naziv elementa može sadržavati slova, brojeve i druge znakove, ali ne može započeti brojem ili znakom zareza, te slovima xml (ili XML ili Xml i sl.); isto tako, ne smije sadržavati ni prazne prostore.[14] Nadalje, XML elementi mogu sadržavati atribute. Atributi su predviđeni da sadrže podatke koji su vezani uz specifičan element, te njihova vrijednost uvijek mora biti navedena unutar navodnih znakova ili jednostrukih navodnika.



Slika 2. XML element s atributom.

[12] A Gentle Introduction to XML. URL: https://www.tei-c.org/release/doc/tei-p5-doc/en/html/SG.html (2019-08-20)
[13] Bosančić, B. Označavanje teksta starih knjiga na hrvatskom jeziku pomoću TEI standarda [doktorski rad]. Zadar: Odjel za informacijske znanosti, 2011. str. 96-98. URL: https://www.bib.irb.hr/561774 (2019-08-20)
[14] W3Schools.com: XML tutorial. URL: https://www.w3schools.com/xml/ (2019-08-20)

Prilikom odlučivanja o tome hoće li element sadržavati atribut ili ne, potrebno je uzeti u obzir činjenicu da atributi, za razliku od elemenata, ne mogu sadržavati više različitih vrijednosti; isto tako, ne mogu posjedovati strukturu stabla i ne mogu se jednostavno proširiti u svrhu budućih izmjena. Atributi se uglavnom koriste za označavanje metapodataka, a elementi za označavanje primarnog teksta.[15] Prilikom izgradnje XML dokumenta, potrebno je držati se strogo definiranih sintaktičkih pravila:

- XML dokument mora i može imati samo jedan korijenski (engl. *root*) element u kojem su pravilno ugniježđeni svi drugi elementi.
- XML elementi moraju imati početnu i završnu oznaku.
- XML elementi su osjetljivi na mala i velika slova.
- XML elementi moraju biti propisno ugniježđeni (engl. *nested*).
- Vrijednost atributa XML elementa mora biti navedena u navodnim znakovima.[16]

Kada govorimo o XML-u, važno je spomenuti *XML Schemu*. *XML Schema* predstavlja način definiranja strukture XML dokumenata, te služi kao alternativa DTD-ju, od kojeg je svakako moćnija zbog toga što su *XML Scheme* pisane u XML-u, te nije potrebno učiti novi jezik. Osim toga, mogu se proširivati i otvorene su izmjenama te podržavaju imenske prostore i heterogene vrste podataka.[17]

[15] Bosančić, B. Uvod u XML. Označiteljski jezici za prikaz i opis sadržaja. Osijek, 2017. [Predavanje]
[16] Isto.
[17] W3Schools.com: XML Schema tutorial. URL: http://www.w3schools.com/schema/ (2019-08-20)

3. PROGRAMSKI JEZICI

3.1. Povijest i razvoj programskih jezika

Početak razvoja programskih jezika vezan je uz izum diferencijalnog stroja Charlesea Babbagea još 1822. godine, kojem su bile potrebne upute za izvršavanje određenih zadataka. No, važnijim se čini njegov izum analitičkog stroja 1830-ih koji je sadržavao većinu ključnih dijelova a koja posjeduju i moderna računala, kao što su središnji procesor, memorija, upute za izvršavanje operacija i organizacija podataka.[18] Daljnji razvoj programskih jezika pratimo kroz razvoj jezika koji su se u početku odnosili samo na niz koraka unošenja naredbi u računalo i njihova izvršavanja. Ovakav jednostavan proces zamijenili su jezici koji su imali naprednije mogućnosti poput logičkog grananja i orijentacije objekata. Od začetka razvoja suvremenih programskih jezika, točnije u zadnjih 60 godina, programske jezike možemo podijeliti u dvije skupine; prva se odnosi na programske jezike koji više nisu u upotrebi a druga na programske jezike koji su i danas u upotrebi. Sljedeći važan korak u razvoju programskih jezika bio je razvoj dvaju koncepata Johna Von Neumanna, koji je razvio, tzv. „*shared-program technique*“. Uz ovu tehniku, razvio je i, tzv. „*conditional control transfer*“, koji je doveo do pojma potprograma. Još jedan važan izum bilo je grananje kôda na temelju logičkih izjava IF, THEN i FOR. Krajem 40-ih godina prošlog stoljeća, pojavljuje se *Short Code*, prvi programski jezik za elektroničke uređaje, a 1952. Grace Hopper razvija prvi kompajler (engl. *compiler*). Kompajler je program koji prepoznaje izjave napisane u jeziku i pretvara ih u nule i jedinice kako bi ih računalo moglo razumjeti.[19] Prijelaz između razvoja prvih i drugih programskih jezika obilježio je razvitak *C* jezika 1972. godine iz *Pascal*a. C jezik koristi sve značajke *Pascal*a, međutim, cilj njegova stvaranja bila je brzina i snaga, ali nauštrb čitljivosti kôda. Unatoč tomu, svejedno je u vrlo kratkom vremenskom razdoblju postao korištenijim programskim jezikom od *Pascala*. Nedugo zatim, nastaje nova metoda programiranja koja je označila prekretnicu, a to je objektno orijentirano programiranje (engl. *object oriented programming*) ili OOP, koje se zasniva na korištenju objekata kao dijelova podataka koje programer može sastaviti i s njima upravljati. Na temelju OOP-a razvijaju se dodaci C jeziku koji ubrzo nakon toga dovode do razvoja C++ programskog jezika koji je objavljen 1983. Mogućnosti koje C++ donosi su korištenje snage koje nudi C uz dodatak objektno orijentiranog programiranja i zadržavanja brzine koju C posjeduje. C++ također se temelji na enkapsulaciji

[18] Henderson, H. Encyclopedia of Computer Science and Technology: Revised Edition. New York: Infobase Publishing, 2009., str. 35-36. URL: https://www.e-reading.club/bookreader.php/135785/Henderson_-_Encyclopedia_of_Computer_Science_and_Technology.pdf (2019-08-21)
[19] Usp. Isto, str. 231.

(engl. *encapsulation*), nasljednosti (engl. *inheritance*) i polimorfizmu (engl. *polymorphism)* kao i drugi programski jezici te generacije.[20]

Važno je spomenuti i jezik *Java* kao još jedan od striktno objektno orijentiranih jezika, koja implementira bitne tehnike kao što su prenosivost kôda i sakupljanje smeća (engl. *garbage collection*). Jedan je od najkorištenijih jezika danas, a koristi se u mnoštvu svakodnevnih aplikacija i uređaja, od mobilnih telefona do navigacijskih sustava i sl.[21] Od novijih jezika, najbitniji je C# programski jezik koji se pokreće na .NET platformi (engl. *framework*), koji je oko 2000. godine razvila ekipa na čelu s Andersom Hejlsbergom i koji se zasniva na C, C++ te sintaksi *Java* programskog jezika. C# izvrsna je implementacija objektno orijentirane paradigme, a kao i u C++ jeziku, to uključuje enkapsulaciju, nasljednost i polimorfizam. Enkapsulacija predstavlja oblik zaštite te razdvaja vanjsko ponašanje od unutarnjih detalja implementacije. Ovakav pristup je iznimno koristan u slučajevima gdje je potrebno višestruko nasljeđivanje, također je izrazito efikasan u onemogućavanju propuštanja memorije jer sadrži velike sigurnosne kopije memorije. Nasljednost označava mogućnost klase da naslijedi neku drugu klasu u svrhu proširenja ili prilagođavanja naslijeđene klase. Nasljeđivanjem iz neke klase se omogućava korištenje funkcionalnosti iz te klase umjesto da se isti kôd ponovo piše. Polimorfizam predstavlja mogućnost da se jedan objekt može pojaviti u više oblika i izdanja. Osnovna ideja polimorfizma je da postoji bazna klasa, koju nasljeđuje više različitih klasa koje mogu izvršavati iste funkcije svaka na svoj način.[22]

3.2. PHP

3.2.1. Osnovno o PHP-u

PHP (*Hypertext Preprocessor*) programski je jezik otvorenog kôda koji se koristi za razvijanje mrežnih stranica i aplikacija, te spada u grupu tzv. pozadinskih tehnologija (engl. *back-end*).[23] Začetak PHP-a pratimo od 1995., kada je Rasmus Lerdorf razvio njegovog prethodnika PHP/FI (*Personal Home Page / Forms Interpreter*). Svrha njegova nastanka bila je izrada skupova skripti koje su napravljene kako bi stvoritelj navedenog jezika mogao pratiti koliko korisnika pristupa njegovom online životopisu, a taj skup skripti nazvao je *Personal Home Page Tool*. S

[20] Bjarne Stroustrup's official homepage. URL: http://www.stroustrup.com/bs_faq.html (2019-08-21)
[21] Oracle: The History of Java Technology. URL: https://www.oracle.com/technetwork/java/javase/overview/javahistory-index-198355.html (2019-08-21)
[22] Albahari, J., Albahari, B. C# 5.0 in a Nutshell: The Definitive Reference. Sebastopol: O'Reilly, 2012. str. 1-3. URL: http://bedford-computing.co.uk/learning/wp-content/uploads/2015/10/Oreilly.Csharp.5.0.in_.a.Nutshell.5th.Edition.June_.2012.pdf (2019-08-21)
[23] PHP: What is PHP?. URL: https://www.php.net/manual/en/intro-whatis.php (2019-08-25)

vremenom se korištenje razvijalo, a samim time rasli su i zahtjevi za funkcionalnošću te je tvorac PHP/FI-ja odlučio objaviti ga kako bi ga i drugi korisnici mogli koristiti, ali i kako bi mu drugi mogli pomoći u ispravljanju grešaka u razvoju. Nasljednik prve verzije bila je druga verzija koja se pojavila 1997. godine te je već tada bila vrlo popularna i imala više tisuća sljedbenika. Bez obzira na više od 50 tisuća domena koje su imale instalirane drugu inačicu PHP skriptnog jezika i više ljudi koji su sudjelovali u razvoju početnog kôda, zapravo je cijeli koncept bio veliki projekt jednog čovjeka.

Nakon prvotne dvije verzije, Andi Gutmans i Zeev Suraski razvijaju PHP 3, verziju PHP-a kakvog danas poznajemo i koristimo. Za razliku od druge verzije koja je bila nadopunjavanje prve, treća verzija napisana je u potpunosti iz početka, ali je objavljena kao nasljednik PHP/FI 2.0. Najveća prednost ove verzije nad prethodnima korištenje je ekstenzija koje se koriste za proširenje osnovne funkcionalnosti PHP-a a što je novim programerima omogućilo pristup razvoju ekstenzija, što je značilo da ne moraju pisati cijeli kôd iz početka, već samo ekstenziju za ono što im je bilo potrebno. Uz navedeno, još jedna od velikih prednosti PHP 3.0 verzije je uvođenje objektno orijentirane sintakse. Funkcionalnost ove opcije očituje se u podatku da je do kraja 1998. PHP 3 koristilo 10% svih svjetskih web poslužitelja. Sljedeća verzija PHP 4 u suštini je ponovo prepisana jezgra sustava u svrhu omogućavanja još veće povezanosti s novim ekstenzijama i bolje performanse PHP aplikacija. Jezgra je nazvana *Zend Engine* i predstavljena je 1999. Pet godina nakon toga predstavljen je PHP 5 čija je nova jezgra *Zend Engine 2.0* objedinjavala novi modul za objektno orijentirano programiranje i nove funkcije.[24] Sljedeća značajna verzija odnosila se na PHP 7 koji izlazi tijekom 2014. i 2015. te sa sobom donosi ogromne pomake u funkcionalnosti i performansi samog jezika. Mukotrpnim radom na refaktoriranju baznog kôda PHP-a, programeri su uspjeli uvelike smanjiti potrošnju memorije i povećati performanse aplikacija, što je vidljivo u rezultatima testiranja, i koji pokazuju da je PHP 7 čak i dvostruko brži od verzije 5.6, a u pojedinim slučajevima i još brži.[25]

3.2.2. Laravel

Laravel je besplatno razvojno okruženje otvorenog kôda za PHP jezik, kojeg je 2011. objavio njegov izumitelj Taylor Otwell. Laravel je prvenstveno služio za popravljanje grešaka i nedostataka koje su se nalazile u razvojnom okruženju *CodeIgniter*, koji se pak pojavio 2006.

[24] Božajić, I. Uvod u PHP i objektno orijentirano programiranje : priručnik za polaznike. Zagreb: Algebra, 2011. str. 4-5. URL: https://www.algebra.hr/cjelozivotno-obrazovanje/wp-content/uploads/sites/3/2017/12/PHP-i-MySQL.pdf (2019-08-25)
[25] PHP: 7.0 New features. URL: https://www.php.net/manual/en/migration70.new-features.php (2019-08-25)

te ga se najčešće isticalo po brzini u odnosu na druga razvojna okruženja PHP-a. Samo razvojno okruženje temelji se na MVC (*Model-View-Controller*) obrascu softverske arhitekture, a kreiranjem novog projekta samo okruženje stvara „kostur“ projekta, odnosno potrebnu MVC strukturu te pripadajuće direktorije i datoteke, što omogućuje korisnu uštedu vremena. Kreiranje projekata, kao i korištenje mnoštva ostalih komandi vrši se putem ugrađenog CLI (engl. *Command line interface*) Artisana.[26]

```
laravel new ProjectName
```

Slika 3. Kreiranje novog projekta pomoću komande Laravelovog Artisana.

Unutar Laravela, komunikacija s bazom podataka se vrši putem njegovoga ugrađenog ORM-a (engl. *Object-Relational Mapper - ORM*) - alata *Eloquent*. Svaka tablica u bazi podataka sadrži svoj odgovarajući model koji se koristi kako bi se moglo s njom komunicirati.[27]

```
$blogPost = new BlogPost();

$blogPostData = [
    'title' = "New blog post",
    'body' = "This is a new blog post about something"
];

$blogPost->create($blogPostData);
```

Slika 4. Primjer stvaranja novog zapisa u bazi podataka kroz $blogPost objekt koji predstavlja BlogPost model.

[26] Laravel: Documentation. URL: https://laravel.com/docs/5.8 (2019-08-26)
[27] Laravel: Eloquent. URL: https://laravel.com/docs/5.8/eloquent (2019-08-26)

4. IZRADA APLIKACIJE TEMELJENE NA PHP-U I XML-u

U ovom dijelu rada opisat će se praktična aplikacija koja je napravljena u sklopu završnog rada. Aplikacija nosi naziv *TestData*, te predstavlja aplikaciju otvorenog kôda koja je bazirana na PHP-u, Laravel razvojnom okruženju za PHP i RPC stilu arhitekture API-ja. Kada je API modeliran prema RPC-u, to znači da krajnje točke prihvaćaju samo GET i POST vrste HTTP zahtjeva. Aplikacija pohranjuje i prikazuje podatke koji se u testne svrhe unose u bazu podataka. Radi se o izmišljenim "osobnim podacima" izmišljenih ljudi koji uključuju informacije o državama i gradovima u kojima žive, sportovima kojima se bave, njihovim avatarima i sl. . Na ovaj način razvojni programeri, tzv. 'developeri', koristeći ovaj API, mogu uštedjeti vrijeme u prvotnim fazama razvoja vlastitih programskih rješenja, jer mogu na brz način dobiti testne podatke iz baze podataka. Za upravljanje promjenama nad izvornim kôdom koristio se *GitHub* na kojem je stvoren repozitorij u kojem se nalazi izvorni kôd aplikacije i u kojem se mogu vidjeti sve promjene kôda.[28] Unutar repozitorija se također nalazi i „*readme*“ datoteka koja sadrži detaljne upute kako postaviti i pokrenuti aplikaciju.

Kao što je navedeno, aplikacija predstavlja API (*Application Programming Interface*)-sučelje za programiranje aplikacija. Jednostavnije rečeno, API omogućuje drugim aplikacijama da komuniciraju međusobno. Najčešće se susrećemo s *RESTful* (*Representational State Transfer*) API-jima koji omogućuju da im se pošalje određeni zahtjev (engl. *request*) odnosno HTTP zahtjevi kao što su GET, PUT, POST i DELETE za nekim informacijama prema određenoj krajnjoj točki (engl. *endpoint*), a koji potom vraćaju klijentu adekvatan odgovor (engl. *response*). Javno dostupne API-je danas posjeduje gotovo svaka velika tvrtka, pogotovo društvene mreže kao što su Twitter[29], Facebook[30], YouTube[31] i sl. Za prikaz odgovora ili podataka koje API dostavlja, najčešće se koriste JSON i XML formati. JSON (*JavaScript Object Notation*) jednostavan je format razmjene podataka neovisan o jeziku. Temelji se na programskom jeziku JavaScript te ga je lako razumjeti i generirati. Što se tiče prikaza podataka, JSON je bolja opcija zbog manje veličine, veće brzine te jednostavnijeg i čitljivijeg kôda.

[28] TestData API. URL: https://github.com/mvavetic/TestData-API (2019-08-26)
[29] Twitter Developer Docs. URL: https://developer.twitter.com/en/docs (2019-08-26)
[30] Facebook Developers Docs: Facebook APIs. URL: https://developers.facebook.com/docs/ (2019-08-26)
[31] YouTube APIs: Google Developers. URL: https://developers.google.com/youtube/ (2019-08-26)

```
{
   "person": {
      "firstName": "Matija",
       "lastName": "Vavetić"
   }
}
```

Slika 5. Primjer JSON kôda

Podržane su sljedeće krajnje točke (*.list* krajnje točke ispisuju sve postojeće zapise, a *.info* ispisuju pojedinačni zapis prema unesenom ID-ju):

```
/api/people.list (GET)
/api/person.info (GET)
/api/person.create (POST)
/api/person.update (POST)
/api/person.delete (POST)

/api/countries.list (GET)

/api/cities.list (GET)
/api/city.info (GET)
/api/city.create (POST)
/api/city.update (POST)
/api/city.delete (POST)

/api/sports.list (GET)
```

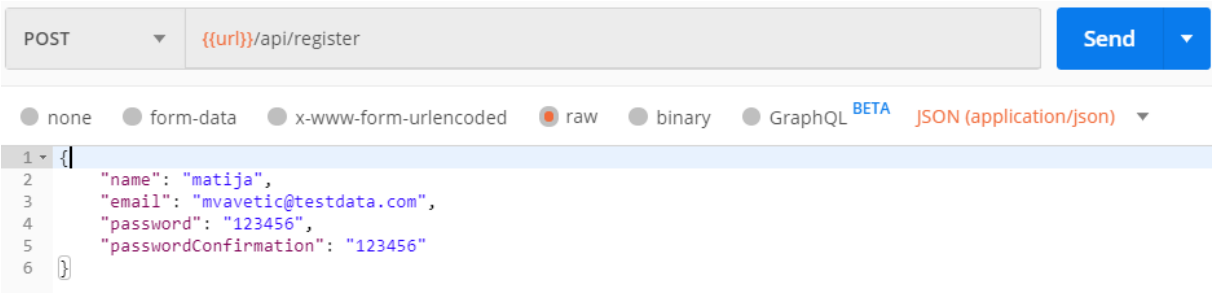
Slika 6. Krajnje točke podržane unutar aplikacije.

4.1. Autentifikacija

Kako bi se osigurala zaštita aplikacije, primijenjen je *Laravel Passport*[32] paket koji koristi *OAuth2* sustav autentifikacije. *OAuth* je otvoreni protokol koji omogućava sigurnu autorizaciju na jednostavan i pouzdan način iz web, mobilnih i desktop aplikacija. Također omogućuje

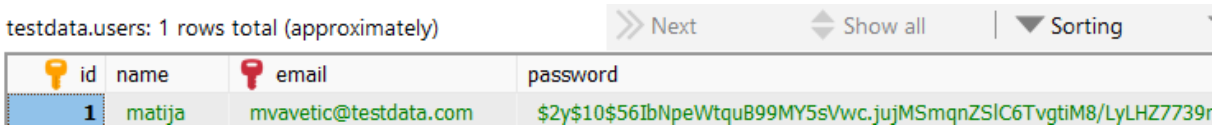
[32] Laravel Passport. URL: https://laravel.com/docs/5.8/passport (2019-08-29)

aplikacijama trećih strana da dobiju vremenski ograničen pristup određenoj web usluzi.[33] Najprije je potrebno provesti registraciju korisnika slanjem zahtjeva s potrebnim podacima (korisničko ime, e-mail adresa, lozinka) na krajnju točku „*/api/register*“.



Slika 7. Primjer zahtjeva koji se šalje prilikom registracije

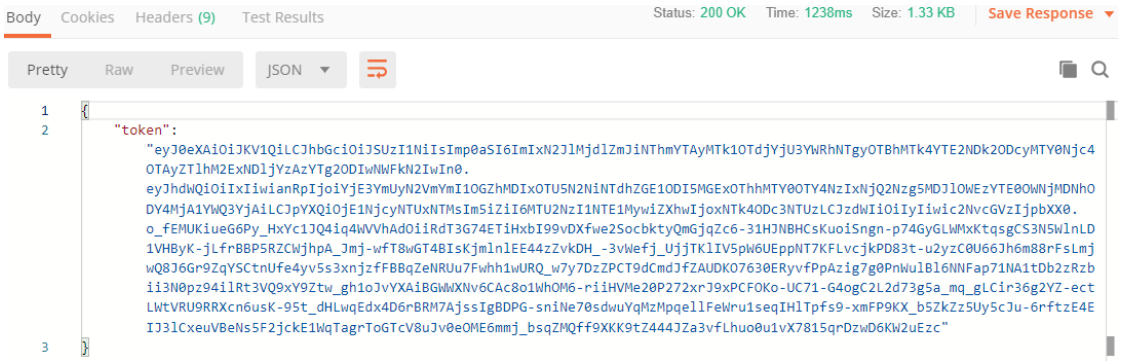
Nakon uspješne registracije, podaci korisnika spremaju se u bazu podataka, a njegova lozinka se kriptira pomoću *bcrypt hashinga*[34], čime se dodatno zaštićuju korisnikovi podaci.



Slika 8. Zapis registriranog korisnika u bazi podataka s kriptiranom lozinkom

Sljedeći korak u ostvarivanju pristupa API-ju je prijava (engl. *login*) pomoću e-mail adrese i lozinke koji se šalju prema krajnjoj točki „*/api/login*“, i ako je prijava uspješna, korisniku se dodjeljuje jedinstveni token koji se veže uz njegov ID i sprema unutar tablice „*oauth_access_tokens*“ dok je prijavljen te se tako ostvaruje pristup API-ju i njegovim krajnjim točkama.

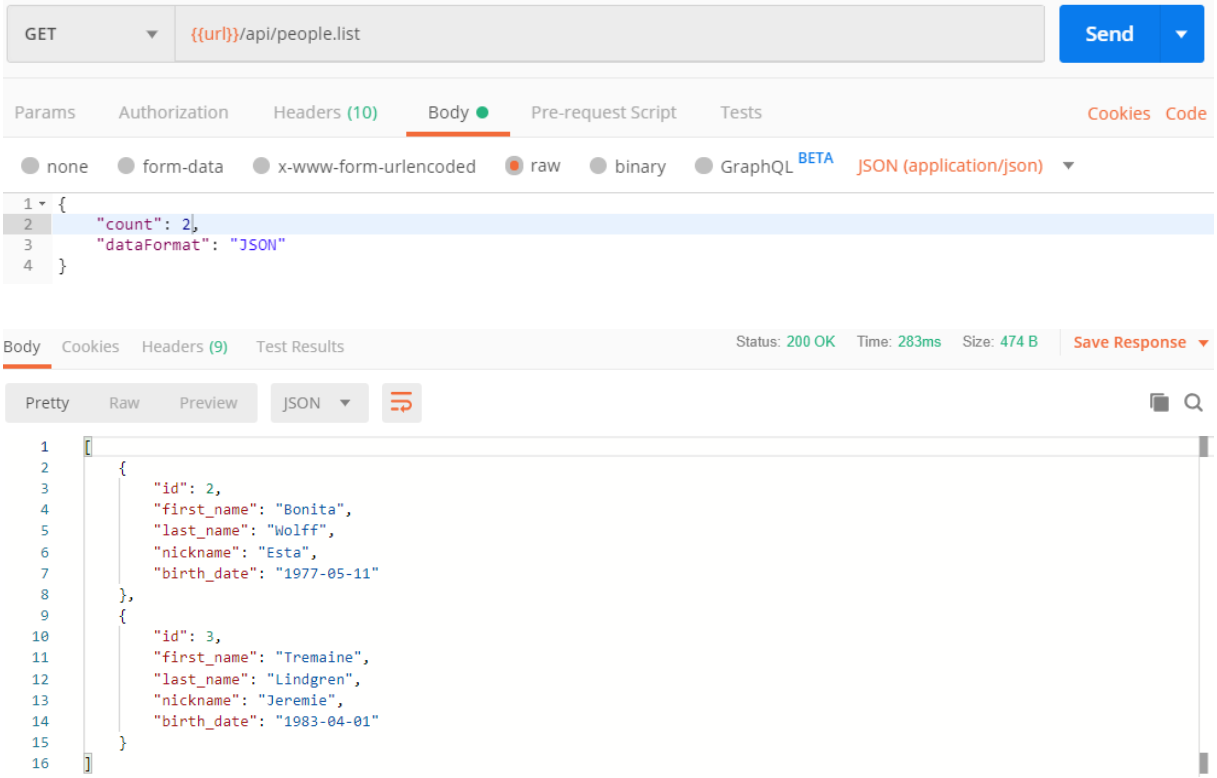
[33] OAuth Community Site. URL: https://oauth.net/ (2019-08-29)
[34] Bcrypt Algorithm. URL: https://www.usenix.org/legacy/publications/library/proceedings/usenix99/full_papers/provos/provos_html/node5.html (2019-09-05)



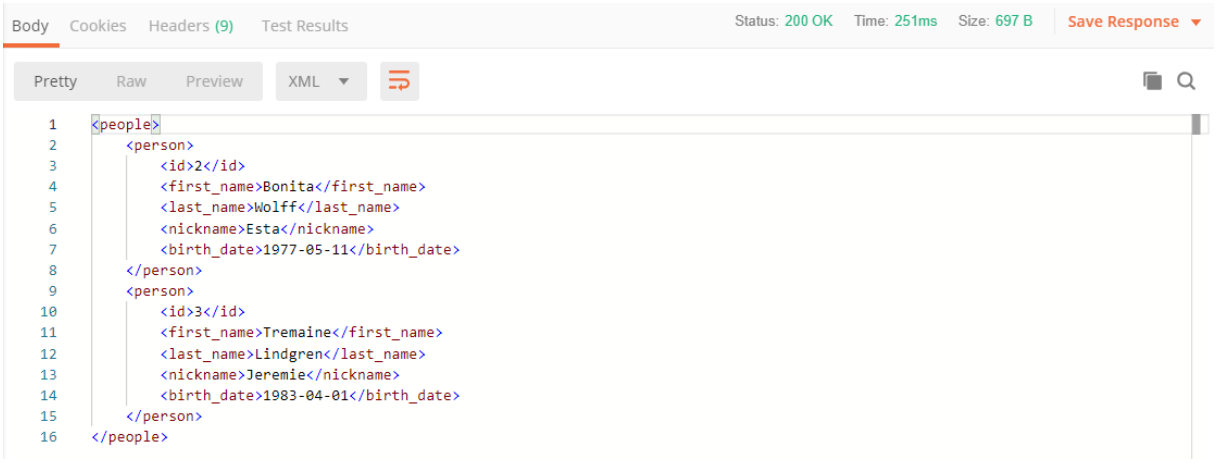
Slika 9. Uspješna prijava i jedinstveni token.

4.2. Upravljanje podacima

Proces dobivanja podataka iz baze podataka je također jednostavan. Šalje se zahtjev s potrebnim podacima nakon čega se, ako je zahtjev ispravan, šalje odgovor s traženim podacima. Također postoji mogućnost biranja između JSON-a i XML-a kao formata u kojem će se podaci prikazati. U sljedećem primjeru prikazan je uspješan proces dobivanja podataka o svim ljudima koji se nalaze u bazi podataka. Zahtjev se šalje na krajnju točku „*/api/people.list*“, te je unutar tijela zahtjeva potrebno unijeti „count“ (broj osoba koje će se prikazati) i „dataFormat“ (format u kojem će se podaci prikazati).

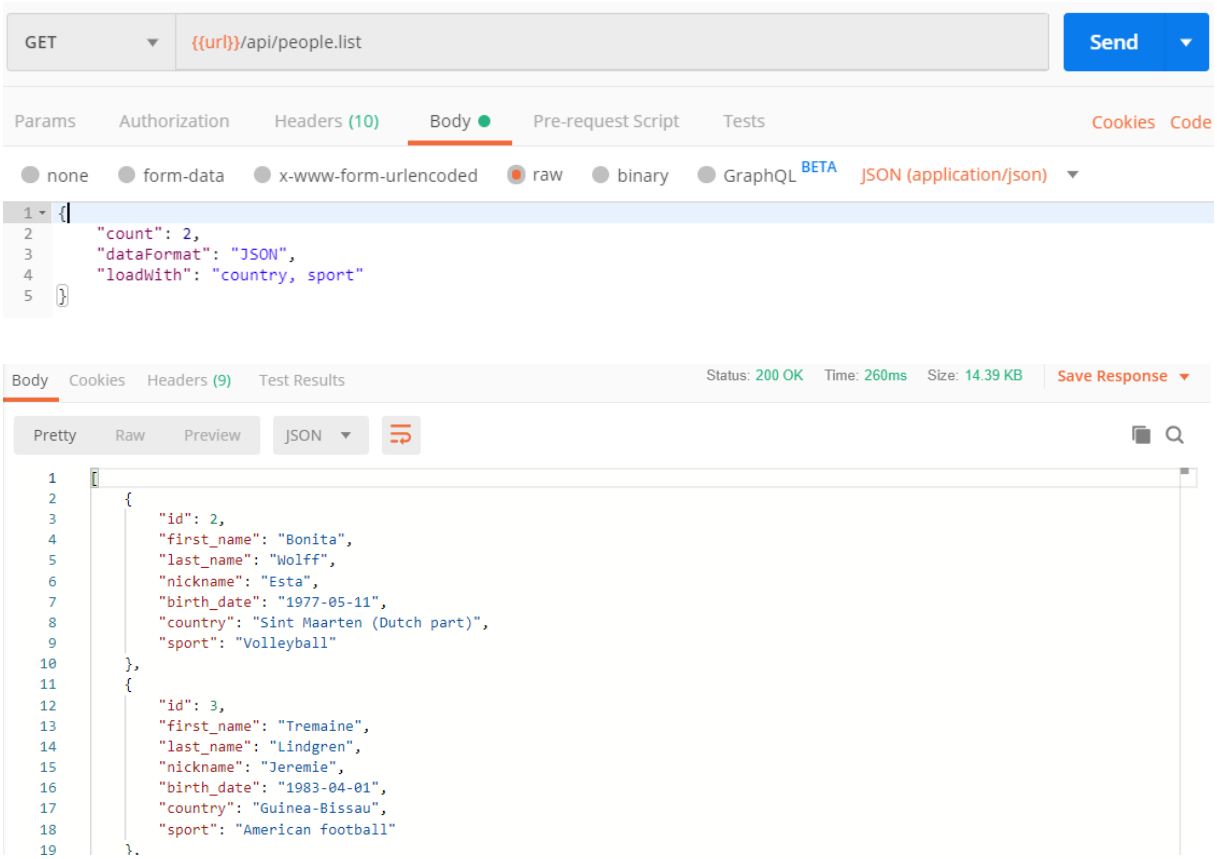


Slika 10. Ispis osoba u JSON formatu.

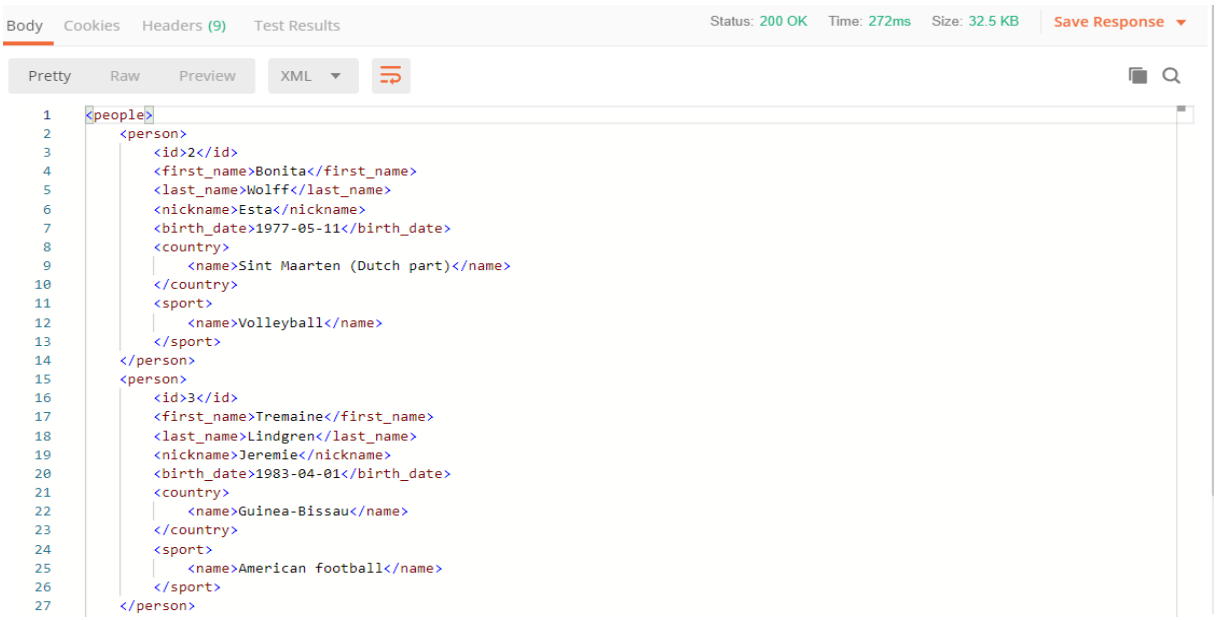


Slika 11. Ispis osoba u XML formatu.

Osim toga moguće je ispisati imena i prezimena osoba s podacima o državi iz koje dolaze te sportom kojim se bave. Za to je potrebno učitati veze pomoću treće opcionalne varijable „*loadWith*“ i unijeti joj vrijednosti „*country*“, „*sport*“ ili obje vrijednosti zajedno.

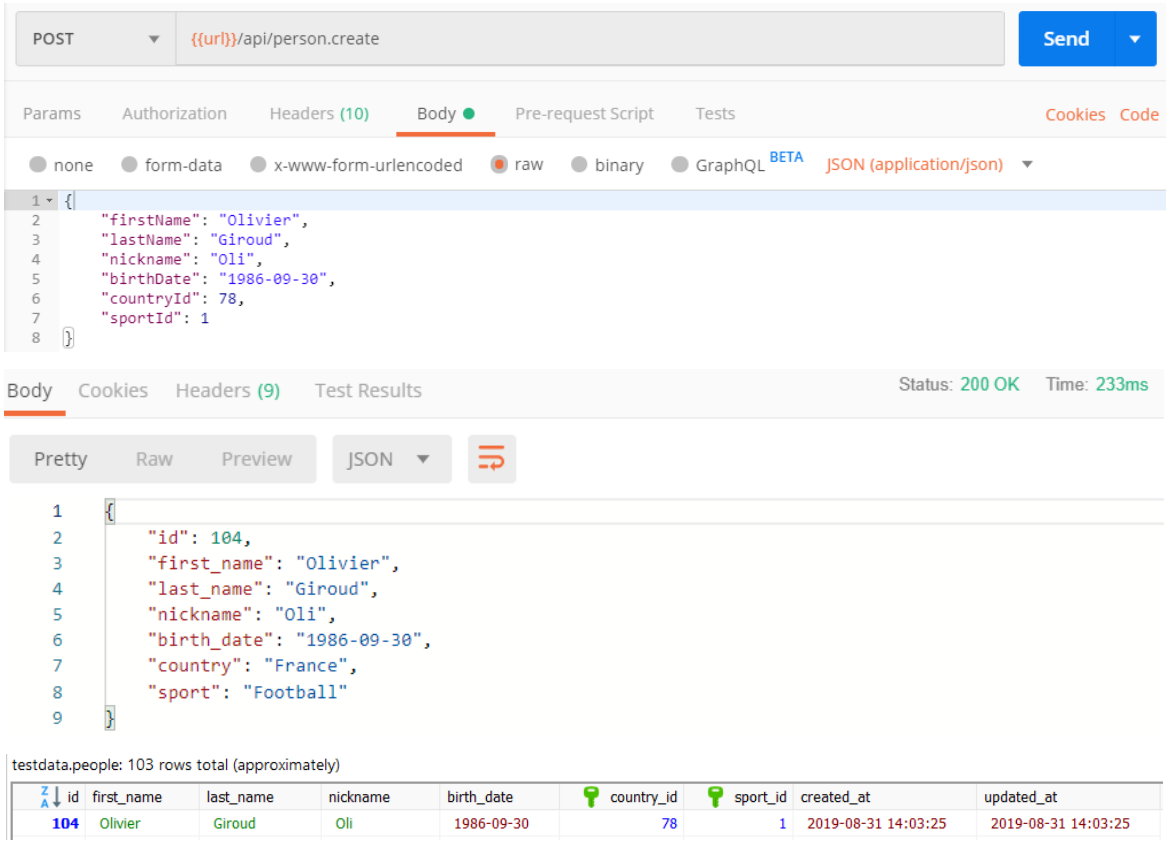


Slika 12. Ispis osoba uz učitanu državu i sport u JSON formatu.



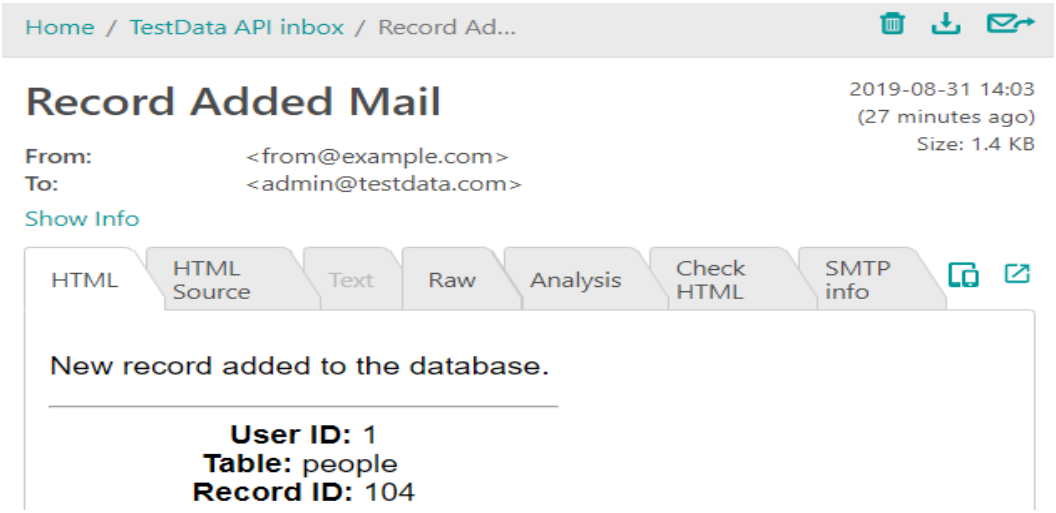
Slika 13. Ispis osoba uz učitanu državu i sport u XML formatu.

Korisnik također ima mogućnost stvaranja, uređivanja i brisanja osoba iz baze podataka slanjem POST zahtjeva prema adekvatnoj krajnjoj točki.



Slika 14. Stvaranje podataka o osobi i novi zapis u bazi podataka.

Kako bi se baza podataka osigurala od neovlaštenog pristupa odnosno neovlaštenog upravljanja podacima, dodan je atribut „*manage*“, koji može poprimiti vrijednosti 0 ili 1, i kojim može upravljati samo administrator sustava. Ukoliko je vrijednost 1, onda korisnik ima mogućnost upravljanja podacima. U svrhu nadzora nad promjenama u bazi podataka, postavljen je automatizam slanja elektroničke pošte administratoru sustava u kojem se nalaze detalji o novom zapisu, te o kojem korisniku, zapisu i tablici se radi. U svrhu testiranja i razvoja ovog automatizma koristio se *Mailtrap*.[35]



Slika 15. E-mail s detaljima novog zapisa osobe u bazi podataka.

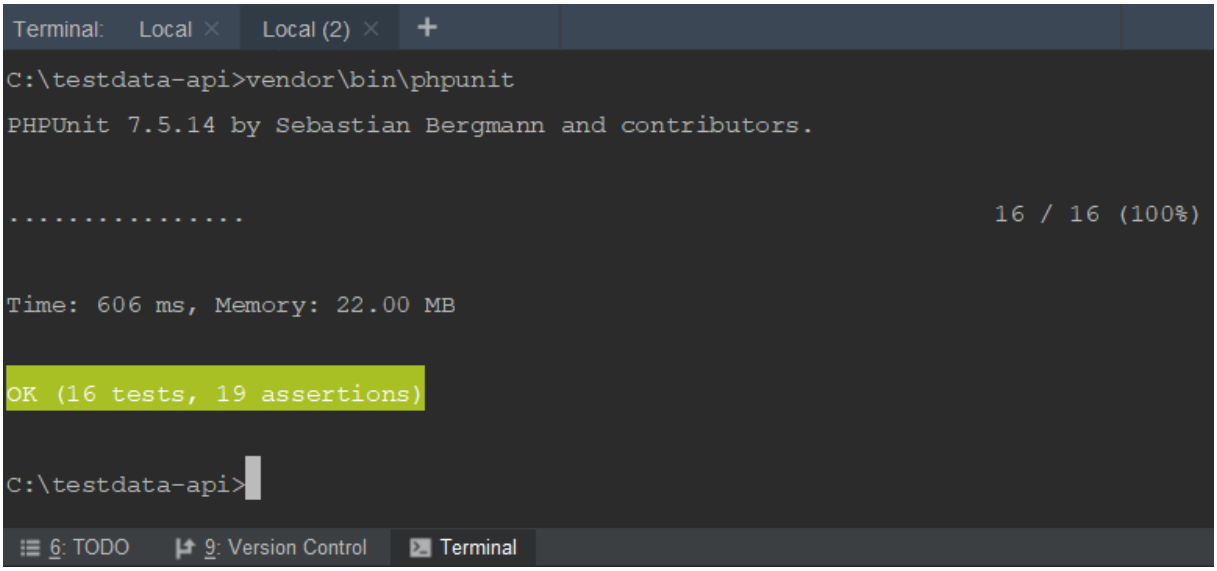
4.3. Testiranje kôda aplikacije

Kako bi se osigurao kvalitetan razvoj aplikacije i njena ispravnost, primijenjena je metoda 'razvoja vođenog testovima' (engl. *test-driven development - TDD*). Ovakva vrsta razvoja aplikacije podrazumijeva često pisanje testova kako bi se osigurala ispravnost kôda. Svrha razvoja aplikacije vođenog testovima je prvo napisati testove, a onda implementirati metode koje će provesti testove. Cilj je proći kroz tri faze testiranja: crvenu (*red*), zelenu (*green*) *i* fazu refaktoriranja (*refactor*). U prvoj fazi test je tek napisan i neće proći provjeru ispravnosti, a nakon uspješne implementacije kôda test prolazi provjeru i nakon toga nastupa refaktoriranje, odnosno čišćenje i unapređivanje kvalitete kôda. Postoji nekoliko različitih vrsta testova, od kojih su najpopularniji tzv. *unit* i integracijski testovi. Tzv. *unit* testovi se koriste za provjeravanje ispravnosti manjih individualnih dijelova kôda, npr. dio pojedine metode, a

[35] Mailtrap.io – dummy smtp email testing. URL: https://mailtrap.io/ (2019-08-29)

integracijski testovi se pišu na kraju kada su svi dijelovi sustava gotovi i kada se trebaju integrirati u cjelinu.[36]

U svrhu testiranja *TestData* aplikacije koristio se *PHPUnit framework* za testiranje koji je napravljen posebno za PHP jezik.[37] Napisano je i nekoliko integracijskih i *unit* testova kako bi se osigurala ispravnost kôda.



Slika 16. Izgled terminala nakon uspješnog prolaza provjere ispravnosti svih testova

[36] Bedeković, H. Razvoj softvera vođen testiranjem [diplomski rad]. Zagreb: Sveučilište u Zagrebu, 2016. str. 7-14. URL: https://repozitorij.pmf.unizg.hr/islandora/object/pmf%3A77 (2019-08-30)
[37] PHPUnit – The PHP Testing Framework. URL: https://phpunit.de/ (2019-08-30)

5. Zaključak

Razvoj računalnih tehnologija donio je niz mogućnosti koje do tada nisu postojale. Kako je ovo područje donijelo niz promjena i novina koje su učinile svakôdnevni život puno jednostavnijim, područje se razvijalo od samoga početka nevjerojatnom brzinom. Isto tako, razvoj novih tehnologija uvelike je utjecao na razvoj društva općenito, a taj razvoj pratimo i danas te se čini da se on odvija još većom brzinom. Rad opisuje razvoj računalnih tehnologija kroz povijest koji je vezan uz razvoj označiteljskih i programskih jezika čije su funkcionalnosti i primjena prikazane u aplikaciji izrađenoj korištenjem navedenih jezika. Navedeni jezici i njihov razvoj označili su revoluciju u korištenju i mogućnostima koje računala i web nude čovječanstvu. Jedan su od najvažnijih faktora koji su omogućili rapidan razvoj informacijske tehnologije u cjelini s obzirom na to da su sa sobom donijeli veliki broj novih mogućosti, kao što je na primjer digitalizacija brojnih, ako ne i gotovo svih, područja ljudskog djelovanja i poslovanja.

Jedan od primjera primjene označiteljskih i programskih jezika je upravo aplikacija izrađena u okviru ovog rada u svrhu prikaza mogućnosti programskih i označiteljskih jezika u upravljanju podacima. Naime, aplikacija je imala za cilj prikazivanje međusobnog i simultanog djelovanja označiteljskih jezika, programskih jezika kao pozadinskih tehnologija na način da se slanjem zahtjeva kroz aplikaciju na određenu krajnju točku dobiju odgovori u XML ili JSON formatu. Također, ovom aplikacijom je prikazana dobra strana API-ja kao vrste sučelja baze podataka koji nudi krajnje točke preko kojih se mogu stvarati, uređivati, brisati i čitati podaci iz iste.

Literatura

1. A Gentle Introduction to XML. URL: https://www.tei-c.org/release/doc/tei-p5-doc/en/html/SG.html (2019-08-20)
2. Albahari, J., Albahari, B. C# 5.0 in a Nutshell: The Definitive Reference. Sebastopol: O'Reilly, 2012. URL: http://bedford-computing.co.uk/learning/wp-content/uploads/2015/10/Oreilly.Csharp.5.0.in_.a.Nutshell.5th.Edition.June_.2012.pdf (2019-08-21)
3. Bedeković, H. Razvoj softvera vođen testiranjem [diplomski rad]. Zagreb: Sveučilište u Zagrebu, 2016. URL: https://repozitorij.pmf.unizg.hr/islandora/object/pmf%3A77 (2019-08-30)
4. Bjarne Stroustrup's official homepage. URL: http://www.stroustrup.com/bs_faq.html (2019-08-21)
5. Blyth, A., Cunliffe, D., Sutherland, I. Security analysis of XML usage and XML parsing. // Computer & Security 22, 6(2003). URL: https://www.sciencedirect.com/science/article/pii/S0167404803006072?via%3Dihub (2019-08-20)
6. Bosančić, B. Povijest označiteljskih jezika. Označiteljski jezici za prikaz i opis sadržaja. Osijek, 2017. [Predavanje]
7. Bosančić, B. Označavanje teksta starih knjiga na hrvatskom jeziku pomoću TEI standarda [doktorski rad]. Zadar: Odjel za informacijske znanosti, 2011. URL: https://www.bib.irb.hr/561774 (2019-08-20)
8. Bosančić, B. Uloga opisnih označiteljskih jezika u razvoju digitalne humanistike. // Libellarium IV, 1(2011). URL: https://hrcak.srce.hr/file/136172 (2018-08-18)
9. Bosančić, B. Uvod u XML. Označiteljski jezici za prikaz i opis sadržaja. Osijek, 2017. [Predavanje]
10. Božajić, I. Uvod u PHP i objektno orijentirano programiranje : priručnik za polaznike. Zagreb: Algebra, 2011. URL: https://www.algebra.hr/cjelozivotno-obrazovanje/wp-content/uploads/sites/3/2017/12/PHP-i-MySQL.pdf (2019-08-25)
11. Coombs, J. H., Renear, A. H., & DeRose, S. J. Markup systems and the future of scholarly text processing. // Communications of the ACM, 30(11), 1987. URL: https://doi.org/10.1145/32206.32209 (2019-08-16)
12. Henderson, H. Encyclopedia of Computer Science and Technology: Revised Edition. New York: Infobase Publishing, 2009. URL: https://www.e-

reading.club/bookreader.php/135785/Henderson_-_Encyclopedia_of_Computer_Science_and_Technology.pdf (2019-08-21)
13. Joloboff, V. Trends and Standards in Document Representation. // Text Processing and Document Manipulation: Proceedings of the International Conference University of Nottingham, 1986.
14. Laravel: Documentation. URL: https://laravel.com/docs/5.8 (2019-08-26)
15. Laravel: Eloquent. URL: https://laravel.com/docs/5.8/eloquent (2019-08-26)
16. Laravel Passport. URL: https://laravel.com/docs/5.8/passport (2019-08-29)
17. Mailtrap.io – dummy smtp email testing. URL: https://mailtrap.io/ (2019-08-29)
18. Oracle : The History of Java Technology. URL: https://www.oracle.com/technetwork/java/javase/overview/javahistory-index-198355.html (2019-08-21)
19. OAuth Community Site. URL: https://oauth.net/ (2019-08-29)
20. PHP: What is PHP?. URL: https://www.php.net/manual/en/intro-whatis.php (2019-08-25)
21. PHP: 7.0 New features. URL: https://www.php.net/manual/en/migration70.new-features.php (2019-08-25)
22. PHPUnit – The PHP Testing Framework. URL: https://phpunit.de/ (2019-08-30)
23. Renear, A. H. Text encoding. // A companion to digital humanities / Susan Schreibman, Raymond George Siemens and John M. Unsworth. Wiley-Blackwell, 2004. URL: http://www.digitalhumanities.org/companion/view?docId=blackwell/9781405103213/9781405103213.xml&chu nk.id=ss1-3-5&toc.depth=1&toc.id=ss1-3-5&brand=default (2019-08-16)
24. Snyder, R. A Practical Introduction to the XML, Extensible Markup Language, By Way of Some Useful Examples. // Proceedings of the 2004 ASCUE Conference. URL: https://files.eric.ed.gov/fulltext/ED490125.pdf (2019-08-20)
25. TestData API. URL: https://github.com/mvavetic/TestData-API (2019-08-26)
26. W3Schools.com: XML tutorial. URL: https://www.w3schools.com/xml/ (2019-08-20)
27. W3Schools.com: XML Schema tutorial. URL: http://www.w3schools.com/schema/ (2019-08-20)